Rapport Projet : Métro c'est trop

Anis BOUZIANE Julien JACQUET Damien DEMONTIS

5 mai 2017

1 Explication des structures de données

2 CODE COMPLET :

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "lecture.h"
#include "mct.h"
#include "dijkstra.h"

TAB initialise_sommets(char *str, TAB M){
  printf("initialise_sommets\n");
  M=init_s(M);
  return initialiser_sommets(str, M);
}

int conversion_sommet_str_sommet_int(TAB M, char
    *sommet_str){
  printf("conversion_sommet_str_sommet_int\n");
  int sommet=-1;
  int i;

  for(i=0; i<NBR_STATIONS;i++){
    if(strcmp(sommet_str,M.TAB[i].nom)==0){
      sommet = atoi(M.TAB[i].index);
      return sommet;
    }
  }
  if(sommet == -1){
    printf("sommet inconnu ou mal orthographi \n");
    printf("ex1: Basilique de Saint-Denis ->
        Basilique_de_Saint-Denis\n");
    printf("ex2: Place d'Italie -> Place_d'Italie\n");
    exit(EXIT_FAILURE);
  }
  return sommet;
}

////////////////////////////////////////////////////////////////////////////////
int main(int argc, char *argv[]) {
  TAB M;
  char *sommet_depart_str;
  char *sommet_arrivee_str;

  if(argc < 3){
    printf("Trop peut d'arguments\n");
    printf("utilisation :\n");
    printf("mct station_de_depart station_d'arrivee\n");
    exit(EXIT_FAILURE);
  }

  if(argc == 3){
```

```
48      sommet_depart_str=argv[1];
49      sommet_arrivee_str=argv[2];
50    }
51
52    if(argc>3){
53      printf("Trop d'arguments\n");
54      printf("utilisation :\n");
55      printf("mct station_de_depart station_d'arrivee\n");
56      exit(EXIT_FAILURE);
57    }
58
59    M=initialise_sommets("metro.txt",M);
60    printf("sommets M initialis s\n");
61    initialise_graph("metro.txt",G,M);
62    printf("graph initialis \n");
63    plus_court_chemin(G,M,conversion_sommet_str_sommet_int(M,sommet_depart_str),conve
64    printf("fin algorithme\n");
65    //printf("showme: %s %s %s\n",
          G[0][238].sm1.nom,G[0][238].sm2.nom,G[0][238].temps);
66    return 0;
67  }
68  ////////////////////////////////////////////////////////////////////////////////
```

../src/mct.c

```c
#ifndef __MCT_H
#define  __MCT_H

#include "constantes.h"
#include "structures.h"
#include "lecture.h"
//STAT initialiser_sommets_et_arcs(char *str);

void calcul_trajet();

void afficher_trajet();

#endif
```

../src/mct.h

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "lecture.h"

TAB init_s(TAB M){ //remplie le tableau des sommet d'un marqueur "UNKNOWN"
  int i;
  for(i=0; i<NBR_STATIONS-1; i++){
    strcpy(M.TAB[i].index,"UNKNOWN");
    strcpy(M.TAB[i].nom,"UNKNOWN");
    strcpy(M.TAB[i].ligne,"UNKNOWN");
    strcpy(M.TAB[i].status,"UNKNOWN");
  }
  return M;
}

void init_g(ARC G[NBR_ARCS][NBR_ARCS]){
  int i;
  int j;
  for(i=0;i<NBR_ARCS;i++){
    for(j=0;j<NBR_ARCS;j++){
      strcpy(G[i][j].sm1.index,"UNKNOWN");
      strcpy(G[i][j].sm1.nom,"UNKNOWN");
      strcpy(G[i][j].sm1.ligne,"UNKNOWN");
      strcpy(G[i][j].sm1.status,"UNKNOWN");
      strcpy(G[i][j].sm2.index,"UNKNOWN");
      strcpy(G[i][j].sm2.nom,"UNKNOWN");
      strcpy(G[i][j].sm2.ligne,"UNKNOWN");
      strcpy(G[i][j].sm2.status,"UNKNOWN");
      strcpy(G[i][j].temps,"99999");

    }
  }
}

TAB initialiser_sommets(char *str, TAB M){ //extrait les donn es de "metro.txt" et les stock dans le tableau des sommets
  int i;
  char type[128];
  char index[128];
  char nom[128];
  char ligne[128];
  char status[128];
  FILE* fic = fopen(str, "r");

  if (fic == NULL) {
    printf("echec ouverture fichier%s\n", str);
    exit(EXIT_FAILURE);
  }

  for(i=0;i<NBR_STATIONS;i++){ //ignore ce qu'il y a apr s la derniere station
```

```
51      fscanf(fic,"%s %s %s %s %s\n", &type[0], &index[0],
            &nom[0], &ligne[0], &status[0]);
52      strcpy(M.TAB[i].index,index);
53      strcpy(M.TAB[i].nom,nom);
54      strcpy(M.TAB[i].ligne,ligne);
55      strcpy(M.TAB[i].status,status);
56    }
57
58  fclose(fic);
59  return M;
60  }
61
62  void associer_graph_data(char *sm1, char *sm2,ARC
        G[NBR_ARCS][NBR_ARCS],TAB M){
63    int i;
64    for(i=0;i<NBR_STATIONS;i++){
65      if(atoi(sm1) == atoi(M.TAB[i].index)){
66        strcpy(G[atoi(sm1)][atoi(sm2)].sm1.index,M.TAB[i].index);
67        strcpy(G[atoi(sm1)][atoi(sm2)].sm1.nom,M.TAB[i].nom);
68        strcpy(G[atoi(sm1)][atoi(sm2)].sm1.ligne,M.TAB[i].ligne);
69        strcpy(G[atoi(sm1)][atoi(sm2)].sm1.status,M.TAB[i].status);
70      }
71      if(atoi(sm2) == atoi(M.TAB[i].index)){
72        strcpy(G[atoi(sm1)][atoi(sm2)].sm2.index,M.TAB[i].index);
73        strcpy(G[atoi(sm1)][atoi(sm2)].sm2.nom,M.TAB[i].nom);
74        strcpy(G[atoi(sm1)][atoi(sm2)].sm2.ligne,M.TAB[i].ligne);
75        strcpy(G[atoi(sm1)][atoi(sm2)].sm2.status,M.TAB[i].status);
76      }
77    }
78  }
79
80  void initialiser_graph(char *str,ARC G[NBR_ARCS][NBR_ARCS],
        TAB M){ //extrait les donn es de "metro.txt" et les
        stock dans le tableau des arcs
81    int i=0;
82    char type[128];
83    char sm1[128];
84    char sm2[128];
85    char temps[128];
86    char garbage[128];
87    FILE* fic = fopen(str, "r");
88
89    if (fic == NULL) {
90      printf("echec ouverture fichier%s\n", str);
91      exit(EXIT_FAILURE);
92    }
93
94    while(fscanf(fic,"%s %s %s %s %s\n", &type[0], &sm1[0],
          &sm2[0], &temps[0], &garbage[0]) != EOF){
95        if(i>=376){ //ignore toute la partie sur les sommets
96          associer_graph_data(sm1,sm2,G,M);
97          associer_graph_data(sm1,sm2,G,M);
98          strcpy(G[atoi(sm1)][atoi(sm2)].temps,temps);
99          i++;
```

```
100         }
101       i++;
102     }
103 fclose(fic);
104 }
105
106 void initialise_graph(char *str,ARC
        G[NBR_ARCS][NBR_ARCS],TAB M){
107   printf("initialise_graph\n");
108   init_g(G);
109   initialiser_graph(str,G,M);
110 }
```

../src/lecture.c

```c
#ifndef __LECTURE_H
#define __LECTURE_H

#include "constantes.h"
#include "structures.h"

TAB init_s(TAB M);
TAB initialiser_sommets(char *str, TAB M);
TAB cleanup(TAB M);

void initialise_graph(char *str, ARC G[NBR_ARCS][NBR_ARCS],
    TAB M);
void init_g(ARC G[NBR_ARCS][NBR_ARCS]);

#endif
```

../src/lecture.h

```c
#include <stdio.h>
#include <stdlib.h>
#include "constantes.h"
#include "structures.h"
#include "string.h"

List fill_start(List l, int a){
  printf("fill_start\n");
  Element* new= malloc(sizeof(Element));
  new->val = a;
  new->next=l;
  l=new;
  return l;
}

void print_list_path(List l, TAB M,ARC
    G[NBR_ARCS][NBR_ARCS],char *temps_total){
  printf("print_list_path \n");
  if(l==NULL || l->next == NULL){
    printf("liste vide\n");
    exit(013);
  }


    while(l->next != NULL){
      printf("en partant de %s allez jusqu'a %s par la
          ligne %s\n",M.TAB[l->val].nom,
          M.TAB[l->next->val].nom,
          G[l->val][l->next->val].sm2.ligne);
    }
}

int conversion_temps_min(int temps_total){
  printf(" conversion_temps_min\n");
  int minutes;
  minutes=temps_total/60;
  return minutes;
}

int conversion_temps_sec(int temps_total){
  printf(" conversion_temps_sec\n");
  int secondes;
  secondes=temps_total%60;
  return secondes;
}



int sommets_tous_traites(int sommets_traites[NBR_STATIONS]){
  int i;
  for(i=0; i<NBR_STATIONS; i++)
    if(sommets_traites[i]==0)
      return 0;
  return 1;
```

```
51 }
52
53 void init_pere(int pere[NBR_ARCS]){
54   printf("init_pere \n");
55   int i;
56   for(i=0; i<NBR_STATIONS; i++)
57     pere[i]=-1;
58 }
59
60
61 void dijkstra(int pere[NBR_STATIONS], int sommet_depart){
62   printf("dijkstra \n");
63   int sommets_traites[NBR_STATIONS]={0};
64
65   int plus_courte_distance[NBR_STATIONS];
66   init_pere(pere);
67
68   sommets_traites[sommet_depart] = 1;
69
70   int i;
71
72   for(i=0; i < NBR_STATIONS; i++){
73     if(atoi(G[sommet_depart][i].temps) != 99999){
74       plus_courte_distance[i]=atoi(G[sommet_depart][i].temps);
75       pere[i]= sommet_depart;
76     }
77     else{
78       plus_courte_distance[i]=99999;
79     }
80   }
81
82   plus_courte_distance[sommet_depart] = 0;
83
84   while(!sommets_tous_traites(sommets_traites)){
85     printf("tours de while\n");
86     //recherche du prochain sommet
87     int min = 99999, a_traiter;
88
89     for(i=0; i<NBR_STATIONS; i++){//pour toutes les stations
90
91       if(sommets_traites[i] == 0){
92         if(plus_courte_distance[i] <= min){
93           a_traiter = i;
94           min=plus_courte_distance[i];
95         }
96       }
97     }
98
99     sommets_traites[a_traiter] = 1;
100
101    for(i=0; i<NBR_STATIONS;i++){
102      if(atoi(G[a_traiter][i].temps) != 99999){
103        if(plus_courte_distance[i] >=
             plus_courte_distance[a_traiter] +
```

```
              atoi(G[a_traiter][i].temps)){
104           plus_courte_distance[i] =
                  plus_courte_distance[a_traiter] +
                  atoi(G[a_traiter][i].temps);
105           pere[i] = a_traiter;
106         }
107       }
108     }
109   }
110 }
111
112 void plus_court_chemin(ARC G[NBR_ARCS][NBR_ARCS],TAB M, int
      sommet_depart, int sommet_arrivee){
113   printf(" plus_court_chemin\n");
114   if(sommet_depart == sommet_arrivee){
115     printf("Vous etes deja arriv ...");
116     exit(1337);
117   }
118
119   int pere[NBR_ARCS];
120   List path = NULL;
121   int temps_total = 0;
122
123   dijkstra(pere,sommet_depart);
124
125   int depart, arrivee = sommet_arrivee;
126
127   path = fill_start(path,arrivee);
128
129   do {
130     depart = pere[arrivee];
131     if(depart == -1){
132       printf("Pas de chemin possible entre %s et
              %s\n",M.TAB[sommet_depart].nom,
              M.TAB[sommet_arrivee].nom);
133       exit(94);
134     }
135     temps_total += atoi(G[depart][arrivee].temps);
136     path=fill_start(path,depart);
137     arrivee=depart;
138   }while(depart != sommet_depart);
139   printf("temps de trajet %d minutes et %d
        secondes\n",conversion_temps_min(temps_total),conversion_temps_sec(temps_total
140
141 }
```

../src/dijkstra.c

```c
#ifndef __DIJKSTRA_H
#define __DIJKSTRA_H

#include "structures.h"
void dijkstra(int pere[NBR_STATIONS], int sommet_depart);

void plus_court_chemin(ARC G[NBR_ARCS][NBR_ARCS],TAB M, int
    sommet_depart, int sommet_arrivee);

#endif
```

../src/dijkstra.h

```c
#ifndef __MCT_H
#define  __MCT_H

struct sommet{
  char index[128];
  char nom[128];
  char ligne[128];
  char status[128];
};
typedef struct sommet SOMMET;

struct tab{
  struct sommet TAB[377];
};
typedef struct tab TAB;

struct arc{
  struct sommet sm1;
  struct sommet sm2;
  char temps[128];
};
typedef struct arc ARC;

ARC G[472][472]; //Matrice d'adjascence

struct element{
  int val;
  struct element* next;
};

typedef struct element Element;
typedef Element* List;

#endif
```

../src/structures.h

```c
#define NBR_STATIONS 376
#define NBR_ARCS 472
#define TAILLE_NOM 200
#define TAILLE_LIGNE 5
```

../src/constantes.h

```
CFLAGS=-c -g  -Wall

all:  mct
  ./mct

compil: mct
  make  mct

test: mct.o lecture.o dijkstra.o
  make  clean
  make  mct.o
  make  lecture.o
  make  dijkstra.o
  gcc -o  tes mct.o lecture.o dijkstra.o
  ./tes Arts_et_M tiers  Galli ni

mct.o:  mct.c mct.h constantes.h  lecture.h structures.h
    dijkstra.c dijkstra.h
  gcc $(CFLAGS) mct.c

lecture.o:  mct.c mct.h constantes.h  lecture.h
    structures.h dijkstra.c dijkstra.h
    gcc $(CFLAGS) lecture.c

dijkstra.o: mct.c mct.h constantes.h lecture.h structures.h
    dijkstra.c dijkstra.h
    gcc $(CFLAGS) dijkstra.c

clean:
  rm  -f  mct
  rm  -f  mct.o
  rm  -f  lecture.o
  rm  -f  dijkstra.o
```

../src/Makefile